

Implementing The Generalised Hybrid Monte-Carlo Algorithm

Z. Sroczynski ^a, S. M. Pickles ^a and S. P. Booth ^b (UKQCD Collaboration)*

^aDepartment of Physics and Astronomy, University of Edinburgh, Edinburgh EH9 3JZ, Scotland

^bEPCC, University of Edinburgh, Edinburgh EH9 3JZ, Scotland

UKQCD's dynamical fermion project uses the Generalised Hybrid Monte Carlo (GHMC) algorithm to generate QCD gauge configurations for a non-perturbatively $O(a)$ improved Wilson action with two degenerate sea-quark flavours. We describe our implementation of the algorithm on the Cray-T3E, concentrating on issues arising from code verification and performance optimisation, such as parameter tuning, reversibility, the effect of precision, the choice of matrix inverter, and the behaviour of different molecular dynamics integration schemes.

1. Introduction

First physics results from the UKQCD dynamical fermions project are presented elsewhere [1]; here we discuss issues arising from the implementation of GHMC [2] in Fortran90 on the Cray-T3E, such as the optimisation of the code to take full advantage of the Cray-T3E architecture, and interesting aspects of the algorithm's behaviour demonstrated in the course of the code verification process.

1.1. Notation

Gauge fields are denoted U and their canonically conjugate momentum fields P . The HMC hamiltonian is

$$\mathcal{H} = T(P) + S_g + S_f \quad (1)$$

where S_g is the plaquette pure gauge action and the fermionic part of the action is

$$S_f = \phi^\dagger (M^\dagger M)^{-1} \phi - 2 \sum_{\text{even } x} \ln \det A_x \quad (2)$$

S_f involves pseudofermion fields ϕ and the odd-even preconditioned fermion matrix

$$M_{xy} = A_{xx} - \kappa^2 D_{xz} A_{zz}^{-1} D_{zy} \quad (3)$$

where D is the usual Wilson hopping matrix and A is the (Sheikoloslami-Wohlert) clover term.

*Presented by Z. Sroczynski

2. Code Improvements

Since our original Fortran implementation, hardware specific code optimisation and algorithmic improvements have seen the CPU time required to complete one trajectory reduced by a factor of 10 to 20, depending on lattice size, β and κ_{sea} .

2.1. Cray-T3E Architectural Factors

Performance on the Cray-T3E is dominated by memory bandwidth. The fact that the processors support multiple instruction issue coupled with the increased complexity of the memory system makes it difficult to take full advantage of the machine's capabilities with even highly optimised Fortran code, so key routines *must* be written in assembler. As a by-product of the tender process, we obtained a set of highly optimised fermion matrix multiplication routines. We have re-written a number of additional routines in assembler and now less than 25% of the run-time is spent executing Fortran code.

Using 32-bit instead of 64-bit floating point numbers to represent the fields improves speed by a factor of 1.7. This must be weighed against degradation of the acceptance rate, the reversibility and the accuracy of calculations of \mathcal{H} (the correctness of the algorithm depends on the accurate computation of the difference of two extensive quantities which are constrained to have nearly identical magnitudes – eventually, *i.e.* as the lattice volume approaches the reciprocal of

the 32-bit machine epsilon, the problem becomes intractable). We have taken special care here, computing energy differences site by site and performing summations in higher precision.

2.2. Algorithmic Improvements

Using BiCGStab [3] instead of Conjugate Gradient to perform the inversions yielded a κ dependent saving of about 40%.

A simple observation which does not seem to be widely known is that the vectors r and s in BiCGStab (in the notation of [3]) can and should be overlapped in memory, saving one vector's worth of storage and two memory copies per iteration.

A further saving of one solve per trajectory can be made by observing that at the start of a trajectory the solution to $M^\dagger Y = \phi$ is known exactly (we initialise the ϕ fields by the heatbath $\phi = M^\dagger \eta$ for Gaussian noise η).

There are special considerations when using the Clover action. A similarity transformation reduces the 12×12 matrix to two 6×6 matrices [4]. This halves the memory requirement and doubles the speed of our inverse multiply routine. Instead of storing A^{-1} explicitly, we store the $L^\dagger DL$ decomposition of A_{ee} ; this saves memory, is just as efficient, more robust and makes the computation of $\det A$ (in S_f) trivial.

Left-preconditioning the BiCGStab by A_{ee}^{-1} reduces the number of iterations required for convergence by some 15%. Using A_{ee}^{-1} as a central preconditioner with CG saves about 25% in terms of iterations.

We do the exponentiation of the conjugate momenta (as required in (5)) *exactly* (up to machine precision). Our first implementation accomplished this by diagonalising the 3 by 3 matrices using library routines. We improved speed by a factor > 6 with a method [5] which exploits the Cayley-Hamilton theorem and requires the solution to a Vandermonde system.

3. Integration Schemes

The numerical integration of U and P forward in “molecular dynamics time” τ can be repre-

sented [6] as an evolution operator $T(d\tau)$

$$T(d\tau) : \begin{pmatrix} U(\tau) \\ P(\tau) \end{pmatrix} \rightarrow \begin{pmatrix} U(\tau + d\tau) \\ P(\tau + d\tau) \end{pmatrix} \quad (4)$$

In general T is composed of two operators; T_U and T_P where

$$T_P(d\tau) : U \rightarrow e^{id\tau P} U \quad (5)$$

$$T_U(d\tau) : P \rightarrow P - id\tau \partial_U (S_g + S_f) \quad (6)$$

Being a numerical integrator, T does not conserve \mathcal{H} exactly but introduces an error $\Delta\mathcal{H}$. It is expected theoretically [7][8] that $\Delta\mathcal{H}$ grows as a power of the timestep $d\tau$. Verifying that this occurs with our code provides a strong test that we have implemented the equations of motion correctly.

We introduce three integration schemes;

$$T_1(d\tau) = T_P(d\tau)T_U(d\tau) \quad (7)$$

$$T_2(d\tau) = T_P\left(\frac{d\tau}{2}\right)T_U(d\tau)T_P\left(\frac{d\tau}{2}\right) \quad (8)$$

$$T_3(d\tau) = T_P\left(\frac{a}{2}d\tau\right)T_U(ad\tau)T_P\left(\frac{a+b}{2}d\tau\right)T_U(bd\tau) \\ T_P\left(\frac{a+b}{2}d\tau\right)T_U(ad\tau)T_P\left(\frac{a}{2}d\tau\right) \quad (9)$$

where $a = 1/(2 - 2^{1/3})$, and $b = -2^{1/3}/(2 - 2^{1/3})$. The expectation is that T_1 , T_2 and T_3 cause $\Delta\mathcal{H}$ to vary as $d\tau^2$, $d\tau^3$ and $d\tau^5$. We find that for some range of $d\tau$ where rounding errors do not dominate, the slopes of plots of $\log \Delta\mathcal{H}$ vs. $\log d\tau$ are respectively 1.982 ± 0.004 , 3.053 ± 0.002 and 5.056 ± 0.006 . Similar results were obtained using just the pure gauge action and the unimproved Wilson action. We can conclude that the equations of motion have been correctly implemented.

The computationally costly part of the the evolution is the evaluation of $\partial_U S_f$ (since it involves the inversion of $M^\dagger M$) in (6), so we split this into

$$T_{pg}(d\tau) : P \rightarrow P - id\tau \partial_U S_g \quad (10)$$

$$T_f(d\tau) : P \rightarrow P - id\tau \partial_U S_f \quad (11)$$

and form the integrator

$$T(d\tau) = T_f\left(\frac{d\tau}{2}\right) \left[T_{pg}\left(\frac{d\tau}{2n}\right) T_P\left(\frac{d\tau}{n}\right) T_{pg}\left(\frac{d\tau}{2n}\right) \right]^n T_f\left(\frac{d\tau}{2}\right) \quad (12)$$

Increasing n means that $\Delta\mathcal{H}$ is reduced without the additional expense of evaluating $\partial_U S_f$ more often. We find that increasing n above 2 does not have a great effect on the acceptance rate, and indeed as the integration demands more computational operations per timestep, the accumulation of rounding errors has a deleterious effect on reversibility and eventually $\Delta\mathcal{H}$. Therefore we use (12) with $n \leq 2$ in production.

4. Estimation of κ_{crit}

Motivated by the idea that a dynamical gauge configuration behaves with some scaling behaviour near a critical point one forms the ansatz

$$N_{CG} \propto \left(\frac{1}{\kappa} - \frac{1}{\kappa_{crit}} \right)^\delta \quad (13)$$

where N_{CG} is the number of CG iterations required to invert $M^\dagger M$ to some given accuracy. Then one expects that a plot of $\log(N_{CG})$ against $\log(1/\kappa - 1/\kappa_{crit})$ would be linear. By using various guesses for κ_{crit} one can estimate its true value as the one that yields a straight line. This is shown in figure 1 for the configurations at $\beta = 5.2$, $\kappa = 0.136$, $c_{SW} = 1.72$ on a $12^3 \times 24$ lattice.

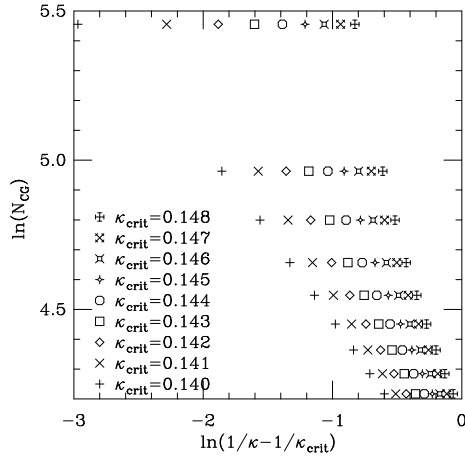


Figure 1. Estimation of κ_{crit} .

The straightest line lies between $\kappa_{crit} = 0.140$ and $\kappa_{crit} = 0.141$. so we can perform a detailed linear fit in this range (the result from subsequent spectroscopy is $0.14033(3)$ [1]). This is a useful technique since many solves are performed in a HMC run (strictly one should only count the first solve of a trajectory, since this is the only one that is guaranteed to be done on a physical configuration) so a statistically significant value for N_{CG} is easily obtained. One then has a quick and reasonable accurate preliminary estimate of κ_{crit} .

5. Autocorrelation Analysis

For the $12^3 \times 24$ lattice at $\beta = 5.2$ we estimate the following for the exponential and integrated plaquette autocorrelation times τ_{exp} and τ_{int} :

κ	0.136	0.137	0.138	0.139	0.1395
τ_{exp}	17	28	40	36	50
τ_{int}	20	34	36	45	64

Acknowledgements

We gratefully acknowledge the support of PPARC in providing funds for the Cray-T3E under grant GR/L22744.

Big shout going out to: J. C. Sexton, B. J. Pendleton and D. S. Henty for helpful advice and discussions, and Bálint Joó for writing some code for us.

REFERENCES

1. M. Talevi; These proceedings.
2. A. D. Kennedy, R. Edwards, H. Mino, B. Pendleton; *Nucl. Phys. (Proc. Suppl.)* **B 47** (1995) 781
3. H. van der Vorst; *SIAM J. Sc. Stat. Comp* **13** (1992) 631
4. M. Göckeler et al.; *Nucl. Phys. (Proc. Suppl.)* **B 53** (1997) 312
5. A. D. Kennedy; Private communication
6. J. C. Sexton, D. H. Weingarten; *Nucl. Phys.* **B 380** (1992) 665
7. R. M. McLachlan, P. Atela; *Nonlinearity* **5** (1992) 1992
8. M. Creutz, A. Gocksch; *Phys. Rev. Lett.* **63** (1989) 9